Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 10 (2012) 634 – 641

Procedia Computer Science

The 9th International Conference on Mobile Web Information Systems (MobiWIS)

Using a Cloud-Centric Middleware to Enable Mobile Hosting of Web Services

Richard K. Lomotey*, Ralph Deters

Department of Computer Science, University of Saskatchewan, 110 Science Place, Saskatoon SK S7N 5C9, Canada

Abstract

The unceasing growth and divergence of the mobile landscape has led to the use of smartphone and tablet devices in consuming Web services in enterprises. However, in heterogeneous Web services, the shift from the mobile client consumer approach to the mobile service hosting approach has received little attention; with no attention given to RESTful mobile services hosting. With the advancement of storage and processing capabilities of these devices; coupled with the high availability of the Web, this paper focuses on the use of the mobile devices as hosts of Web services in an E-health domain. To achieve this goal, the challenges of latency and Web resources state change synchronization has to be addressed to ensure data reliability; since service accessibility is facilitated by the mobile devices which communicate via unstable wireless networks.

In this paper, a cloud-centric middleware technique is employed to enable access to the mobile hosts. The paper presents mobile hosting of light-weight Web services which is deployed in a real world system, and proposes a middleware platform for the update management of Web resources state changes in unreliable wireless networks. Our current implemented project, called *SOPHRA*, which is a joint E-health project with the Geriatrics Ward at the City Hospital in Saskatoon, supports mobile communication over Wi-Fi using HTTP.

Keywords: REST; Mobile Cloud Computing; Mobile Provisioning

1. Introduction

The increasing growth in the use of mobile devices such as smartphones and tablets as reported by Gibbs [1] is shaping enterprise businesses. Today's market reports suggest that there is a promising future for the mobile commerce space though the revenue being generated currently is still insipid compared to

* Richard K. Lomotey. Tel.: +1-306-881-8387; fax: +1-306-966-4884.
E-mail address: richard.lomotey@usask.ca.

doi:10.1016/j.procs.2012.06.081

their ecommerce counterparts [2]. Equally, the use of mobile devices in non-commercial enterprises such as E-health – where healthcare delivery is facilitated by communication tech, has witnessed a phenomenal rise [3].

Furthermore, most of today's smartphones and tablet devices have good support for diverse networks and protocols. Thus, connectivity can be established between mobile participants via 4G, Wi-Fi, and Bluetooth [6]. Additionally, the improvement in mobile connectivity has led to the advancement in mobile cloud computing, where mobile devices consume services from cloud platforms and infrastructures [8].

As the mobile field continues to advance, researchers and developers are no longer just modeling these devices as client consumers especially in heterogeneous Web services, but they are gradually shifting their focus to mobile Web services hosting (also known as mobile provisioning) [6]. The new design paradigm is made possible due to the increasing capabilities of mobile devices in terms of processing power and storage. In a current joint E-health project with the Geriatrics Ward at the City Hospital in Saskatoon, Canada, we have identified a huge potential in mobile hosting of services to support the healthcare professionals to share medical records among themselves. As part of the project, we use mobile devices to aid healthcare professionals to securely access authorized information about patients whose records are hosted on other colleagues' mobile device. In Figure 1, the screenshots of the application, called *SOPHRA*, is shown on an Android powered tablet device. Srirama et al. [6] noted that mobile hosting turns the mobile host into a multi-user node where the owner of the device can work in parallel with other mobile participants who are consuming the services without any interference. From the medical perspective, our project is a huge leap to aiding the healthcare professionals with the information management and convenient accessibility of their patients' record.

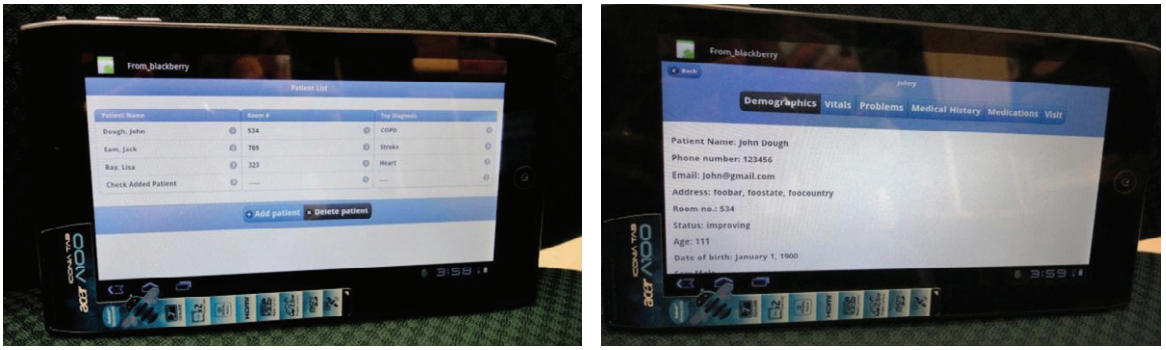

Fig. 1. (a) Homepage of SOPHRA; (b) Demographic record of a particular patient (Due to privacy issues, the data shown is fake)

However, mobile devices primarily communicate via wireless channels which can be unstable due to user mobility and bandwidth fluctuation. Thus, this research has identified the challenges of *latency* and *synchronization of data* (Web resources) among the mobile participants (the healthcare professionals). To address these challenges, we present a cloud-centric middleware framework which aids the healthcare professionals to share their mobile hosted Web services reliably and in soft real-time. As well, the middleware supports both the SOA [16] and REST [12] design paradigms of Web services. Our motive for choosing the cloud infrastructure as a service is because the cloud provides a highly available and fault-tolerant environment for hosting resources [12].

The details of the architecture are explained further in this paper. The remainder of this paper is organized as follows. In Section 2, we expounded on the architectural design of *SOPHRA*. Section 3 presents a description of the implementation and the evaluation of the system. In Section 4, we reviewed related works on mobile hosting and the paper concludes in Section 5 with the summary of our findings and future research.

2. Mobile Host Architecture

This paper presents a mobile host architecture based on how SOPHRA addresses the challenges of resources state change management that arise from hosting services on mobile providers in unreliable wireless networks. A middleware framework is employed on a private internal cloud to enable mobile hosts to be reached via Wi-Fi by other health care professionals. The architectural design as illustrated in Figure 2, is classified into three tiers namely: mobile service requesters, middleware, and mobile service providers/hosts. The middleware is hosted on the cloud infrastructure and the mobile devices connect to the middleware through 802.11g Wi-Fi 54Mbps connection. Also, the implementation took into account both SOAP and RESTful paradigms of building Web Services (WS) for the purposes of evaluating the performance of each WS.

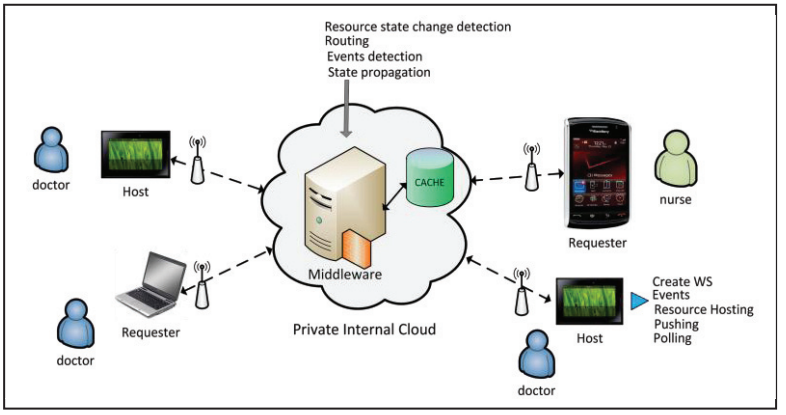

Fig. 2. Mobile Host Architecture

The patient records which are modeled as Web Services (WS) are hosted on the mobile hosts and the records of each host are replicated on the middleware to ensure high availability. In cases involving unreachable health care professionals (hosts), the resources of that host can be accessed on the middleware. There are times when some hosts could be temporarily unavailable due to network loss. As a result, there is the tendency of having outdated information (inconsistencies) on some mobile nodes. As well, the middleware which acts as the providers' backup could be storing outdated information considering the fact that a health care professional (host) could have updated his records but due to connectivity issues, the update might not reflect on the middleware. Thus, we present the best effort which has to do with giving the best data while eventual consistency is achieved. The main functionalities of the various components are described in the next section.

2.1. Middleware Platform/Framework

The cloud-hosted middleware serves as a router for all the asynchronous messaging between the mobile service requesters and the mobile service hosts. All the information of each mobile provider is logged in the middleware cache for callbacks; as a result, the middleware uses pushing to send updates to the mobile participants that have subscribed for services as and when they arrive in soft real-time. The main functionalities of the middleware are discussed below:

- ***Read/Write Request***: The middleware controls the race condition between a read request and a write request that are issued at the same time from the mobile participants. This is achieved by introducing timestamps on each request. The middleware determines which request comes first and processes that request with a higher priority.

- ***Caching***: Since REST affords stateless communication between system components, RESTful resources of the mobile providers are replicated independently in the middleware cache. The cache has a RESTful interface that supports the CRUD methods. This means that the content of the cache can be created, read, updated, and deleted. With an HTTP GET request, the replica resources will be pushed from the cache to the requester. Also, the middleware creates a resource or an update of a resource state through a write request using HTTP POST or PUT respectively. Additionally, the cache stores states of all the mobile providers that have subscribed for services. However, for the SOAP WS version, the acceptable HTTP method is POST for all the operations. Thus, the proprietary middleware cache is modeled to understand whether a particular POST request for a SOAP WS is a read or write request based on the HTTP request headers.
- ***Events and Routing***: All communications are routed through the middleware since some health care professionals can disconnect due to network instability. In such cases, requesters who want to access the resources and services from disconnected mobile hosts can access it from the middleware cache. The middleware is modeled as a reverse proxy by using long-polling technique to detect updates of resources or services on the mobile hosts. If an update is detected, the new state of the resource is pushed to the middleware to update its cache. The middleware also understands events such as:
  - *Resources state change*: When updates arrive in the middleware cache, the new state of resources overrides the older versions. The middleware then propagates the changes to all mobile participants who have subscribed via HTTP.
  - *Connection of a client*: The middleware sends updates to connected mobile nodes as and when they arrive in soft real-time.
  - *Disconnection of a client*: When write requests are sent to disconnected providers, the middleware creates the resources and store them in the cache. The updates are pushed from the middleware to disconnected mobile nodes when they reconnect. Also, a mobile service requester is notified if a provider is disconnected.

2.2. Embedded Browser-Oriented Mobile Framework

The mobile framework consists of the mobile service *requesters* and the mobile service *providers/hosts*. The requesters are the users plus the devices that send HTTP requests to the middleware and the provider either for a data to be created or to fetch an existing data. The mobile service host is the device that hosts/provides the Web Services. Additionally, a mobile service provider can act as a service requester. In our framework, the mobile service hosts behave as Web servers.

Recently, more developers are looking towards HTML5 mobile applications as a solution for targeting multiple devices and platforms. This is because the Web browser is becoming the default platform and its de facto standards are HTML5 and JavaScript [14]. Since mobile devices have an embedded browser, coupled with the advancement of HTML5, we adopted the hybrid app methodology in the design of the mobile framework to build the mobile web app that looks and functions as a native app. With HTML5, the functionalities that native app developers employ to have access to the device can be imported into the mobile Web app. HTML5 also supports server and services which makes it affordable to consume Web services in the embedded browser regardless of the mobile device platform.

Though the mobile providers have common functionalities as the requesters, the former have additional functionalities. The idea is that, the mobile providers will act as a Web server as well as a consumer. However, the Internet Protocol (IP) address of a mobile device changes as the user moves from one hotspot to another. In addition, registering the mobile device name within a network domain is not ideal because the user can move from one network domain to a different domain. In view of these challenges, we implemented the HTML5 server services to establish a dual communication channel between the mobile hosts and the middleware. Thus, the mobile provider can push RESTful resources to the

middleware or poll resources from the middleware. In addition, a mobile requester can send an HTTP request to the provider using the IP address or the registered computer name of the middleware – which is running in the cloud.

When a user sends a write request (e.g. HTTP POST method) from the mobile device, the request is created as a REST-WS resource. The newly created resource is given a globally unique identifier (id) which enables other mobile participants to select or search for that service/resource. The mobile providers' resources are accessed by requesters through URIs provided by the middleware. Also, the responses from the provider make use of the HTTP status codes. This notifies the middleware on the state of a particular resource. The requester is informed whether there is an error in a request (with a 400 status code) or the request is successful (with a 200 status code). Additionally, new updates on the middleware from other health care professionals might not reflect at the providers' side at the same time. To address the problem of keeping outdated resources state, the mobile resources validation is done using HTTP HEAD requests. Since network connectivity is crucial in mobile technology, a periodic HTTP HEAD request is sent to the middleware for only the *Etag* – which is a unique attribute of a Web resource. The mobile nodes compare the Etag to determine whether the local copy of the resource is the same as the incoming resource from the middleware. A change in Etag is an indication of a possible update of an existing resource.

3. Implementation and Evaluation

The mobile platforms that are considered in the implementation are the BlackBerry Playbook, Android tablet devices, and Apple iOS. Since the mobile platforms are heterogeneous in terms of their underlying operating systems, we employed the embedded browser pattern, using CSS, HTML5, and JavaScript, to write a single code base that is deployed on multiple platforms. The BlackBerry version of the application is built as a BlackBerry WebWorks [17] project; which supports multiple web based languages. The same WebWorks code was reused to compile an Android WebView [17] project. The WebView framework supports the deployment of mobile web applications on the Android devices. In addition, the same code is exported to the iOS platform and compiled as an Xcode [17] project. Also, the middleware is implemented using Erlang in order to handle high concurrency and database distributions. The Erlang middleware is built on the Generic Server Behaviour (gen_server) process which is a module that enables the implementation of a server for the request-response interaction.

To evaluate the performance of the system, we deployed various experimental setups that simulate the accessibility of resources on the mobile hosts as well as the scalability of the middleware platform for both the REST and SOAP services. The BlackBerry Playbook simulator which is running as the requester and the Android 4.0 tablet emulator which is running as the provider are deployed on different computers with the following specifications: *Processor: Intel Core i5, CPU 650 @ 3.20GHz 3.19GHz, RAM: 4GB (2.99GB usable), System 32-bit OS*. The Erlang middleware is hosted on the internally owned private cloud platform - with the following specifications: *Windows 7 Enterprise, Service Pack 1, Processor: Intel(R) Xeon(R), CPU E5140 @ 2.33GHz 2.33GHz (2 processors), Installed memory (RAM): 16.00GB, System type: 64-bit OS.* Each experiment is repeated five times and as shown in Figure 3 and Figure 4, the minimum, average, and maximum time in each round of testing is plotted.

The round trip time between the requester and the host was evaluated based on concurrent requesters that each executed 25 read requests and the result is illustrated in Figure 3. Figure 4 shows the outcome of how the middleware responds to requesters when the intended host is unreachable. Also, Figure 5 shows the performance of the overall system with regards to SOAP Web Services accessibility since SOAP is verbose compared to REST. Furthermore, the scalability of the middleware was evaluated based on throughput for 500 mobile participants, as illustrated in Figure 6, who issue concurrent requests from 500

to 40000 and the average throughput was found to be 320.42 requests/second for the RESTful Web Services.

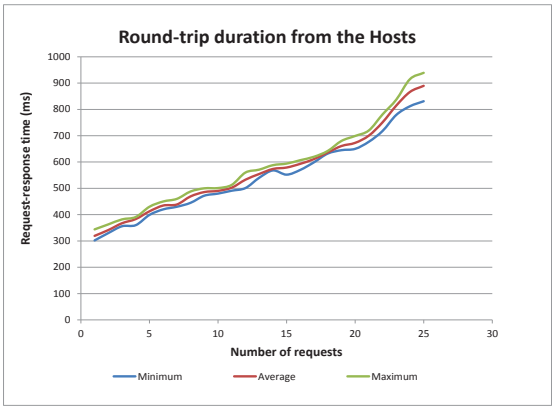

Fig. 3. Round trip duration from the mobile host

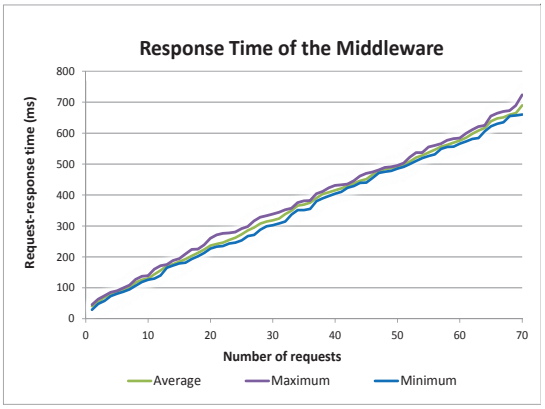

Fig. 4. Middleware's performance in case the mobile host is unavailable

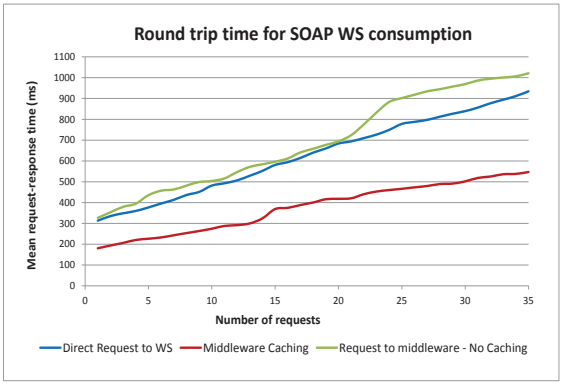

Fig. 5. SOAP WS Accessibility

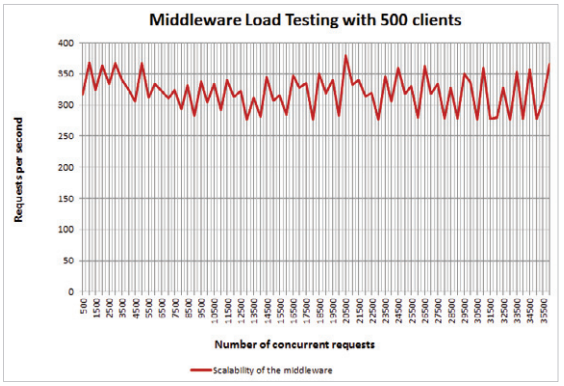

Fig.6. Middleware load testing with 500 clients

From the tests, it is observed that there is a significant boost in system performance considering the fact that it takes far less time to access resources with the cached middleware component. Furthermore, it is observed that the request error rate is zero percent; which suggests that the cloud platform is highly available. Also, we used other metrics such as read and write mixed requests to determine the inconsistency window of our system. It was found that it takes approximately 90.11ms for 500 newly created Web services on a mobile host to become visible on all mobile participants that connect for services. As well, as shown in Figure 5, though SOAP Web services pass verbose XML traffic compared to REST in the findings in [15], our framework handles the traffic management efficiently due to the middleware caching component. In view of the middleware caching, the performance of our system is not impacted even if secure HTTP (i.e. HTTPS) request is sent between system components. This is as a result of the fact that even if requests are not cached in the browser, the middleware still pushes data to the requester very fast. Another improvement of our system is the extreme scalability of the middleware. While the total number of healthcare professionals who will be using SOPHRA is anticipated to be 200 in the near future, the middleware supports 500 users with ease as observed from the test result.

4. Related Literature

4.1. WS vs. REST*

Pautasso et al. [4] compared WS-* which they described as "Big" Web Services, with RESTful Web Services. Their paper concluded that the best design architecture depends on the needs of the developer since the standards have different architectural designs. Also, Wicks et al. [11], noted that SOA focuses on reusability of software and integration as well as strong support for packaging which makes changing of older versions of software very fast and at minimal cost. The SOA architecture normally has a *Services* module which is a platform that is exposed to processing software that controls specific set of tasks [12]. The second constituent is a *Client* module which discovers services that are reusable and engages in a bi-directional message exchanges via Internet or Intranet with the Services module. The *Client* module also uses UDDI and WSDL to discover services and produces SOAP messages [12]. The *Registry* module is a repository where service providers publish services.

Apart from the SOA paradigm, the REpresentational State Transfer (REST) approach has been employed in other research [12, 15]. REST is an architectural principle that uses the Web platform for distributed computing [4, 12]. REST is better understood in the context of identifying everything as a resource, representation, and state. The design follows certain technological principles as described in [9]:

- *Everything is a resource*: All the identifiable entities must be considered as a resource and should be assigned an ID.
- *Identification of resources through URI:* The key resources should be given Universal Resource Identifiers (URIs) which will facilitate interactions within the system.
- *Uniform interface:* Resources can be manipulated through representations using HTTP methods such as GET, HEAD, POST, PUT, DELETE, OPTION, TRACE, PATCH and CONNECT.
- *Self-descriptive messages:* Since resources are decoupled from their representation, it makes content accessibility very simple regardless of the format of the resource content [4].
- *Stateless interactions:* While resources have states, their interactions should be kept stateless.
- *Hypermedia as the engine of application state (HATEOAS):* In order to navigate between resources, URIs such as hypertext can be used in a resource representation. HATEOAS aids the client to know the next steps to take since the returned URI contains links to available options.

4.2. Mobile Hosting

It is surprising that as at now there are only a handful of studies on mobile provisioning. Srirama et al. [7] are one of the pioneering researchers on enabling the mobile device as a provider of Web services. In their initial studies, they proved the feasibility of hosting Web services on the mobile device by adopting the SOAP messaging approach. On issues involving IP addressing in mobile P2P communications, the paper proposed the adoption of two approaches: High-Speed Circuit Switched Data (HSCSD) dial-up connection and General Packet Radio Services (GPRS) environments. Later, the authors extended their work in [6] to facilitate the mobile provider to handle SOAP requests over HTTP.

Also, Meads et al. [5] employ the middleware technique to provide a communication interface for ubiquitous devices to communicate with mobile providers in heterogeneous networks. The paper concludes that mobile providers can be reached via Bluetooth or Wi-Fi, an approach that gives requesters the flexibility to explore different communication channels. Furthermore, Hassan et al [8] researched on managing the limited resources of the mobile provider and proposed a mobile web service partitioning scheme. As a result, complex business processes can be executed by the mobile provider with a backend super computer doing most of the high demanding computations. Additionally, Aijaz et al. [15] demonstrated the high performance of RESTful mobile web provisioning compared to SOAP services.

5. Conclusion and Future Works

This paper is the first to present a mobile provisioning framework, called *SOPHRA*, for both SOAP services and RESTful Web services that is deployed in a real world system. The designed framework employs cloud-centric middleware which enables healthcare professionals at the Geriatrics Ward in the City Hospital in Saskatoon, Canada, to reliably host medical records on their mobile devices in an unreliable Wi-Fi environment. It is encouraging to know from the evaluation of *SOPHRA* that in mobile multi-node systems, it takes fractions of a second for health care professionals to have access to data from a mobile provider/host. Also, in the deployment of this project, security and privacy of the patient data was considered since these requirements are paramount in the E-health domain. But how we handled the enforcement of these requirements is not elaborated in this paper.

The next version of *SOPHRA* which will be released soon will extend on the current version and consider autonomic computing. This approach will aid the system to be more resilient and have fault recovery components for better performance in situations of system failures.

References

[1] Gibbs, C. The Rise of Tablets in the Enterprise. GigaOM Pro, June 2011.

[2] Warren, C. Native App vs. Web App: Which Is Better for Mobile Commerce?, Available online at: http://mashable.com/2011/05/23/mobile-commerce-apps/

[3] Ranck, J. The Rise of Mobile Health Apps. October 2010.

[4] Pautasso, C., Olaf, Z., and Leymann, F. “RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision,” WWW'08, p 805-813, 2008.

[5] Meads, A., Roughton, A., Warren, I., and Weerasinghe, T. “Mobile Service Provisioning Middleware for Multihomed Devices,” Proceeding WIMOB '09, Networking and Communications IEEE Computer Society Washington, DC, USA 2009.

[6] Srirama, S.N., Jarke, M., and Prinz, W. “Mobile Web Service Provisioning. Telecommunications,” (AICT-ICIW '06).

[7] Srirama, S.N., Jarke, M., and Prinz, W.“Mobile Host: A feasibility analysis of mobile Web Service provisioning,” 4th International Workshop on Ubiquitous Mobile Information and Collaboration Systems, UMICS 2006, a CAiSE'06 workshop.

[8] Hassan, M., Weiliang, Z., and Yang, Y. “Provisioning Web Services from Resource Constrained Mobile Devices,” Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on 5-10 July 2010.

[9] Feng, X., Shen, J., and Fan, Y. “REST：An Alternative to RPC for Web Services Architecture,” First International Conference on Future Information Networks, ICFIN 2009, p 7-10, 2009.

[10] Christensen, J. “Using RESTful web-services and cloud computing to create next generation mobile applications,” Proceedings of OOPSLA, p 627-633, 2009.

[11] Wicks, G., Van Aerschot, E., Badreddin, O., Kubein, K., Lo, K., and Steele, D. “Powering SOA Solutions with IMS," Pg. 9 Publisher: IBM Redbooks Pub., Date: March 30, 2009, Part Number: SG24-7662-00, Pages in Print Edition: 410.

[12] Wang, Q., and Deters, R. "SOA's Last Mile-Connecting Smartphones to the Service Cloud," cloud, pp.80-87, 2009 IEEE International Conference on Cloud Computing, 2009.

[14] Monash, C. “DBMS2: Read-your-writes (RYW), aka immediate, consistency,” A Monash Research Publication, May 1, 2010. Available: http://www.dbms2.com/2010/05/01/ryw-read-your-writes-consistency/

[14] Pearce, J. “HTML5 and the Dawn of Rich Mobile Web Applications,” InfoQ Sections: Development, Architecture & Design, Jun 24, 2011, Available: http://www.infoq.com/presentations/HTML5-Dawn-of-Rich-Mobile-Web-Applications

[15] Aijaz, F., Ali, S. Z., Chaudhary, M. A., and Walke, B. “Enabling High Performance Mobile Web Services Provisioning,” Published in: Vehicular Technology Conference Fall (VTC 2009-Fall), 2009 IEEE 70th, 20-23 Sept. 2009.

[16] Siddiqui, B. ‘Deploying Web Services with WSDL: Part 1. IBM Developer Works,’ 1 Nov 2001, Last accessed: January 15, 2012. http://www.ibm.com/developerworks/library/ws-intwsdl/.

[17] PhoneGapWiki. Available: http://wiki.phonegap.com/w/page/33313613/Changelog WebView.